International Journal of Advanced Robotics and Intelligent Systems
Volume 1, Issue 01, 2025

System-level Approaches to Power Efficiency in FPGA-based Designs (Data Reduction Algorithms Case Study)

Arcos[1], F. Pérezjh[2]
[1]Ciencias Básicas, Universidad Autónoma Metropolitana
[2]Electrónica, Universidad Autónoma Metropolitana, México

Research Article

Date of Submission: 03-04-2025 **Date of Acceptance: 6-05-2025** **Date of Publication: 09-07-2025**

Abstract:

This study presents first findings from a system-level investigation of FPGA-based systems' power efficiency. Sophisticated systems (such embedded sensor nodes) may be implemented thanks to advanced FPGA chips. It makes sense to move the design process to higher abstraction layers, or system-levels of design, since creating such sophisticated applications is unaffordable at lower levels. At these higher abstractions, our work demonstrates that at least a certain degree of power awareness is possible. A approach for a power-aware, system-level algorithm paritioning is described, along with first findings. Due to the significance of data reduction methods in wireless sensor networks (WSNs), we have chosen them as the case study. The given concepts are expected to be applicable to various untethered embedded systems based on FPGAs and other comparable programmable devices, even if the study has been concentrated on WSN applications of FPGA.

1. Introduction

An important example of an embedded system is a sensor node. For both military and civilian purposes, a typical sensor node consists of a processor unit, a sensing unit, a power unit, and a wireless communication unit [1]–[3]. In a field, the sensor node's power supplies are often few or non-replaceable. The performance and power consumption of sensor nodes are severely limited by this issue. Because of their sophisticated power and energy management, processing devices with fixed architecture, such as digital signal processors (DSP) or microcontrollers (MCU), continue to be the most often used method for sensor node implementations. Advanced multimillion-gate reconfigurable designs, such as Altera and Xilinx FPGA devices, are vastly more powerful, but [4], [5]. As a result, reconfigurable designs, such as software-based processors (LatticeMico, Nios, MicroBlaze, PicoBlaze, XTensa), have received more attention lately [4]–[7]. While there are currently only a few wireless sensor node applications that use FPGA chips, and in these applications, FPGA is primarily used as a supporting processing unit (e.g. [8]-[11]), it is anticipated that more flexible sensor nodes that can adapt to a wider range of scenarios (including unpredictable ones) will be developed by using such reconfigurable processing units. A typical FPGA has many hard cores, such as memory blocks, digital clock controllers, encryption circuits, and custom multipliers, in addition to the primary array of slices and I/O blocks [12]. Configurable interconnections and switching structures—which are necessary to accomplish the programmability of FPGA—increase loads and, therefore, power consumption, even if the power and performance of FPGA are often compared to application-specific integrated circuits (ASIC), for example [12], [13]. This is a limitation of FPGA, hence for systems based on FPGA, a thorough examination of power characteristics is very crucial. In addition to robust FPGA devices, sophisticated high-level design tools are available to enable the rapid synthesis and prototyping of complex processing units using FPGAs or other complex devices. These tools include compilers (Quartus, ISE), hardware description languages (Verilog, VHDL), system-level hardware description languages (e.g., Han- del-C or Catapult C), etc. In contrast to laborious but effective low-level approaches, system-level techniques are not power-aware, which means that significant power and/or hardware overheads may be included in high-level designs, such as those developed at [4], [5], and [15]. This paper's primary goal is to demonstrate that system-level design methodologies may integrate a certain degree of power awareness with almost little overhead. Using the findings of several tests on power optimization in FPGA designs, we illustrate it. The DK Design Suite and the Handel-C language are used for the experiments. The case study of data reduction methods yields the findings. This decision was made on purpose since one of the main problems with wireless sensor networks (and other related fields) is data minimization.

The paper's subsequent parts are organized as follows: A summary of FPGA power consumption and techniques for power estimate and reduction is provided in Section 2. Selected data reduction strategies used in untethered embedded systems (such sensor nodes) are presented in Section 3. The experimental findings achieved for such methods are described in Section 4, the main body of the study. First, we show that system-level representations of the hardware and power characteristics of FPGA designs are sufficiently accurate. We next use this insight to

demonstrate how system-level design partitioning might result in power savings. Although we concentrate on algorithms that are divided into parallel-running domains, Section 5 offers some concepts for sequentially executable domains. The paper is concluded in Section 6.

2. PowerConsumptioninFPGA

2.1. DynamicandStaticPower

Static and dynamic components make up the power consumption of CMOS devices, such as FPGA [12], [13], [16]–[19]. Signal switching at the device transistors is the source of CMOS devices' dynamic power consumption [16], [18], and [19]. It is clear that the clock frequency and signal switching frequencies are connected. Therefore, a multi-resource system's dynamic power consumption is often described as follows:including adjustments to the clock frequency, supply voltage, and capacitance) could provide some benefits. For instance, on-chip memories are advised over off-chip memory due to the high capacitance of external conditions [19]. Tight scheduling limitations, such as [12], [18], may also reduce the capacitance, requiring place-and-route algorithms to choose resources with lower capacitance. The most efficient way to reduce power usage (a quadratic term in (1) and (2)) is to lower the supply voltage. Lower supply voltages, however, cause circuit delays, which impair performance, hence it must

$$P = \sum_i C_i \cdot V_i^2 \cdot f_i$$

becarefullybalancedagainstanyperformancedrop.De- creasing the clock frequency can also reduce power where Ci, Vi, and fi stand for the ith resource's capacity, voltage swing, and clock frequency, respectively (e.g. [12], [13], [18], [19]).

The complexity of the implemented design clearly determines the real dynamic power of FPGA devices. The effective capacitance of resources, resource consumption, and resource switching activity are the design-dependent elements that affect dynamic power [12], [13], and [20]. The total of the parasitic capacitances of linked wires and transistors is known as the effective capacitance. It is clear from the resource consumption that FPGA offers more resources than are often needed to carry out a certain design (unused resources do not waste the dynamic power). The average number of signal transitions throughout a clock cycle is known as the switching activity. It usually has something to do with the clock frequency, although it may also be influenced by other things, such the timing of the input signals. Thus, (1) may be reformulated as follows: sumption.However, it could need design modifications, particularly for devices operating under preset timing limitations.

2.3.PowerConsumptionEstimationinFPGA

Real measurements or simulation-based estimations may be used to get specifics on FPGA power usage [12], [13], [20–22]. The most precise power information is provided by actual measurements, such as [20], but the measured device has to be representative. Although the simulation-based method to power estimations is more practical, the findings are simply approximations. Most power estimating methods now in use rely on the switching capacitance and corresponding variables like average resource use and average switching activity, as shown in [12], [13], [17]-[19], [20], [22], and [23]. Since the majority of implemented designs are synchronous and controlled by system clocks, these methods are appropriate for estimating the power consumption of FPGA devices.

3. Data Reduction Algorithms

Where Ui and Si stand for the use and switching of certain resources, respectively. The leakage current between the power source and the ground is the primary cause of the static power consumption. According to [19], the leakage current is dominated by the sub-threshold leakage current, which is dependent on temperature and the threshold voltage Vth. Studies like [12] demonstrate that, depending on the temperature, clock frequency, and implemented design, the static power of contemporary FPGAs, such the Virtex-II series (SRAM-based FPGA, 0.15 µm technology), may vary from 5 to 20% of the total dissipated power. However, we do not address static power concerns in this study since the static power of FPGA is mostly technology-dependent and does not vary with design complexity.

2.2. Reduction of Power Consumption

There are three methods for reducing FPGA dynamic power usage, per [20]. First, adjustments may be made at the system level, such as by changing the algorithms that are used. Second, a designer may alter the logic partitioning, mapping, placement, and routing if the FPGA design is predetermined. If such adjustments are not possible, enhanced operating circumstances (this Overview It goes without saying that increasingly sophisticated embedded systems must handle more data. When a lot of data has to be sent wirelessly, handling it becomes considerably more challenging. According to some studies, the cost of transmitting a single bit of data across a certain distance may equal the cost of executing three thousand CPU instructions locally (see [24]–[27]). As a result, data reduction, or compression, becomes a critical problem. There are two types of data reduction (compression) algorithms: lossless and lossy, for example [28], [29]. Applications that cannot accept any discrepancy between the original and decompressed data utilize lossless approaches. Loss-less compression methods often create a statistical model of the data and use that model to translate the data to bit strings. By allowing for distortion during the reconstruction process, lossy compression algorithms provide a much higher compression ratio. Generally speaking, lossy compression techniques use one or more suitable basis functions to convert provided data into a new data space [30]. Different criteria, such as relative complexity, memory needs, CPU speed requirements, compression ratio, and distortion level, may be used to assess compression algorithms (see [28], [29], and [31]).Wireless Sensor Network Data Reduction Wireless sensor network applications do not often employ data reduction. Memory footprints and subpar process unit performance are the main drawbacks (see [31]–[33]). As a result, it is not recommended to employ common loss-less data compression methods such as LZO, BZIP2, PPMd, and other PC-based algorithms (see [31]). However, certain efforts on similar algorithms are used in sensor nodes with restricted performance and power (e.g. LZW in [34]). Huffman and RLE coding are two other lossless data reduction algorithms used in sensor networks [35]. Certain pre-processing methods, such as altering data descriptions and raising the compression ratio, are also often used [34, 35]. These utilize decorrelation techniques like Wavelet Transform (WT) to characterize data structures before to using Huffman codes, and Burrow-Wheeler Transform (BWT) and Structured Transpose (ST) to reorganize data prior to LZW coding. However, when lossy transformations are used, the latter introduces certain distortions. Novel compression strategies have been created for wireless sensor networks in order to get around the drawbacks of conventional algorithms [25], [27], [31], [32], [36]-[40]. These include differential coding lossless techniques, pipelined in-network compression, coding by ordering, and simple low-complexity video compression schemes like JPEG with a few tweaks.

In wireless sensor networks, lossy data reduction involves approximation and aggregation [24], [27], and [40]. Aggregation provides a summary of the data at regular intervals in the form of basic statistics, such as average, maximum, minimum, etc. This method works well for lowering the amount of data, but it is not very suitable for applications that need precise historical data, like monitoring and surveillance. When there is a significant amount of redundancy in the data, approximations are used, such as histograms, wavelets, discrete cosine transform, linear regression, etc. In wire-less sensor networks, there are other techniques for reducing data, such as [26], [32], [41], and [42]. They are not a topic of our discussion since they include distributed processing and integrate routing, data fu-sion, and data aggregation. There may be further needs for data reduction methods in wireless sensor networks, even while the shear data volume is the main problem and immediately affects the communication capacity, for example [26], [34]. Data reduction strategies, for instance, are intended to lower communication

latency, improve energy efficiency (by lowering the energy used for data transmission, [25]), and lower overall energy consumption, [26]. All things considered, one of the primary concerns for data reduction techniques in wireless sensor networks is energy awareness, either directly or indirectly. Consequently, the case study for this research is data reduction algorithms.

4. Experiments

This section covers the testing on the FPGA implementation of two data reduction techniques. The main objective of these investigations is to show that dynamic power awareness at the system level may be achieved using FPGA systems. The algorithms that are selected are Huffman coding and arithmetic coding. This case study, however, provides an illustration of what we believe to be a workable approach with wide applicability, since further testing has shown that similar results have been obtained for alternative algorithms with varied topologies.

5. Because of its simplicity of use, minimal hardware and performance requirements, and the kind of data that has to be sent or stored, Huffman coding is a common method for embedded systems [35]. In the worst scenario, however, issues might occur if the source data is only binary (in-network data such as tracking, detection, and classification), has very skewed probabilities, or has an alphabet that is too small. [28], [29]. By creating the expanded alphabet, which consists of symbols arranged in blocks of two or more, this issue may be partly resolved. However, this causes the codebook to increase exponentially. Unlike Huffman coding, which generates codes for any sequence of that length, arithmetic coding gives code-words to specific sequences [28], [29]. But putting arithmetic coding into practice takes a lot more effort. Consequently, while arithmetic coding is a viable option, it has not yet been used to wireless sensor networks as far as we are aware. However, our findings imply that arithmetic coding might be a viable

6. 8. Choice for FPGA-based applications. Dividing a design into many domains is a tried-and-true technique. Although further compression and decompressor separation are also discussed later, it is obvious that the selected algorithms may be separated into compressors and decompressors. For system-level design power consumption optimization, we use such decompositions as the main technique [43]. It is shown that by selecting the right clock frequencies for each region and using the right partitioning algorithms, significant power savings may be obtained without looking at the hardware-level details of the designs. A feature-rich design suite is the DK Design Suite. The algorithms are implemented at the system level in Handel-C using an environment for C-based algorithmic design input, simulation, and synthesis. To analyze the power efficiency of the designs, we build the designs for the RC203 development board, which is equipped with a Xilinx Virtex-II FPGA (xc2v3000fg676-4). One of the Xilinx Integrated Software Environment (ISE) accessories, XPower, is used to monitor power usage at the hardware level. XPower provides the estimates using simulated data that depicts the activity of the implemented system. Sensor nodes and associated equipment are often positioned in unpredictable environments. Consequently, we make the arbitrary assumption that the activity rate of the implemented designs is 50%. To lessen the differences between XPower estimates and the actual hardware implementations, we decided to use the external FPGA pins as the direct data inputs and outputs.

6.1. Methodology

vBoth clock frequency and design size have an impact on dynamic power consumption, as shown by equations (1) and (2). However thecomplexity of FPGA de At various levels, indicators are assessed differently. The number of comparable NAND gates represents the system-level (abstract) complexity. Therefore, the abstract complexity of a particular algorithm is fixed

(based on the structure of the algorithm and the efficiency of the compiler). The total complexity of the algorithm is just the sum of the complexities of its separate components, even if it is broken down into several portions.

Therefore, in certain non-defined units (NDU), the system-level "dynamic power consumption" of a design may be expressed as a product of the clock frequency and the corresponding NAND number. The translation of system-level structures (netlists) onto FPGA resources (slices, I/O blocks, interconnections, etc.) determines the hardware-level complexity. However, the mapping results might fluctuate greatly depending on the clock frequency, particularly for decomposed algorithms that can be physically divided into many hardware domains or implemented inside a single domain. As a result, the primary query for system-level power estimates is whether algorithmic abstract complexity—that is, the quantity of comparable NAND gates—can serve as a trustworthy indicator of the real dynamic power consumption. Although it seems sense that power increases with the amount of NAND gates, it's crucial to assess fluctuations brought on by variations in mapping and placement (for example, utilizing different clock frequencies), single-domain vs multi-domain implementations of a deconstructed algorithm, etc. The experimental validation of this problem is presented in Section 4.2, which confirms the viability of our approach, i.e., that we can use the system-level complexity of designs to fairly accurately represent the power characteristics of the actual FPGA implementations, as we have not found such validation in the available sources.

6.2. System-level Power Estimate Accuracy In this experiment, we investigate the hardware-level performance properties of a deconstructed algorithm that is run at various clock rates (the decomposed Huffman coding is actually selected as the compressor and decompressor).ccuracy of System-level Power Estimates In this experiment, we examine the hardware-level performance characteristics of a deconstructed algorithm that is operated at different clock speeds (the compressor and decompressor are essentially the decomposed Huffman coding).has knowledge of one or two different topics. We do not use any chip area restrictions and permit map, place, and route tools to do unrestricted optimizations in order to prevent any distortion of the findings. The compressor and decompressor have almost equal system-level complexity, although having different internal architectures, therefore the system-level dynamic power estimates would be similar. Huffman coding was used for this reason. Although they are implemented in two different clock domains, the compressor and decompressor in Design A are located within the same module. Each of them is implemented in a distinct single-domain module in Design B. These values are basically irrelevant, even if Huffman coding was applied to data with a width of 1 bit, an alphabet of 2 elements, and a sample size of 32 items. Many variations of these ideas have been implemented in hardware with different clock frequencies (the lowest and maximum clock frequencies are determined by the platform limitations). We anticipate that the hardware-level power estimations will not change despite the inevitable differences in the implementation's physical layouts. While Figures 1 through 4 show example designs that are, as expected, varied, Tables 1 through 3 include the equivalent hardware-level estimates of dynamic power computed using XPower. Comparing rows 1, 2, and 3 of Tables 1 and 2 to rows 1 and 4 of Table 3 reveals that the additional dynamic power consumption of the compressor and de-compressor implemented independently is roughly equal to the total dynamic power consumption for the design with both compressor and de-compressor. The differences fall below the 5% cutoff. As predicted by the system-level computations, the data also demonstrate that the dynamic power demand varies according to the clock frequency shift. The design's power characteristics don't change, despite the implementations' different physical layouts (see Figures 1-4).

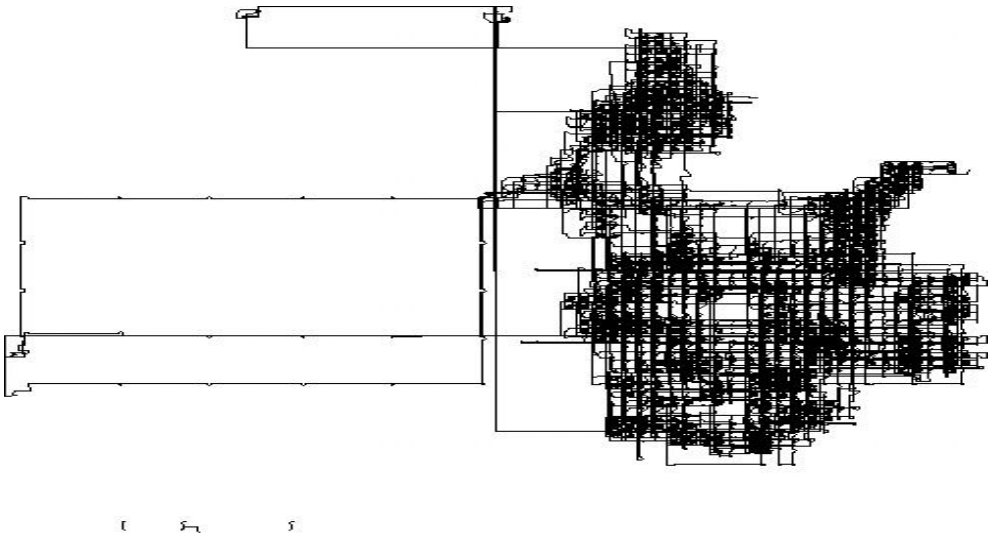

Fig.2.DesignBwithonlycompressor(15MHz) – Huffman coding.

Table1.Onlydecompressor–DesignB.

AlgorithmPartitioningintoParallel Domains–Approach

Inthesubsequentexperiments,weusealgorithmpar- titioningasatoolforpowerreduction.Thesamealgo-rithms,i.e.

Huffmancoding

Clock frequency [MHz]	Total dynamic power (clock+logic+signals)[mW]
6	1.57+5.31+13.66=20.54
15	1.04+13.06+33.73=47.83
24	1.67+20.89+54.46=77.02

andArithmeticcoding Table2.Onlycompressor–

DesignB.

Clock frequency [MHz]	Total dynamic power (clock+logic+signals)[mW]
6	1.04+5.06+12.43=18.53
15	1.04+12.43+30.91=44.38
24	1.67+19.88+49.40=70.95

Table3.Theoverallpowerconsumption(decompressor/ compressor)–DesignA.

Decompressor clock frequency [MHz]	Compressor clock frequency [MHz]	Total dynamic power (clock+logic+signals) [mW]
6	24	2.77+25.19+63.07=91.03
8	22	2.82+25.25+64.00=92.07
12	18	2.57+25.40+64.98=92.95
15	15	1.97+25.48+65.79=93.24
18	12	2.48+25.65+68.03=96.16
22	8	2.48+25.84+67.71=96.03
24	6	2.53+25.95+65.75=94.23

The experiment's findings are significant for the system-level clock domain algorithm partitioning that was previously discussed. They verify that the anticipated clock frequency and the abstract complexity of the designs (such as the number of equivalent NAND gates) may be used to estimate power consumption at the system level. This approach can be used to determine the best partitioning strategies and/or the ideal clock frequencies for different domains, even though we are unable to estimate the absolute values (which rely on the conversion ratio from non-defined units (NDU) to milliwatts; this should be done separately for a given model of FPGA).

The case study is their compressors and decompressors, in fact. We start by concentrating on splitting into concurrently running tasks (the alternative case is covered in short in Section 5). Both methods' compressors and decompressors use the partitioning strategy. The compressor and decompressor are separated into two domains that operate concurrently (see Subsection 4.4 for further information). The most power-efficient clock frequencies are suggested for the domains based on their system-level features. The following are specifics of the system-level study of the designs:

System-levelhardwarecomplexity

The DK Design Suite, a system-level algorithm, is first constructed and synthesized at the net-list level. The hardware complexity (resources) at the system level is determined by the corresponding number of NAND gates in the architecture. Clearly, these outcomes are platform-independent. Even if the synthesized designs are eventually targeted to appropriate hardware (using Xilinx ISE software), the resources are only evaluated at the system level. The domain is synthesized and compiled separately at the system level to determine the corresponding number of NAND gates when it is detached from an algorithm. The complexity (i.e., the equivalent number of NAND gates) of the remaining algorithm may be readily computed by subtracting the number of gates in the isolated domain from the overall number of gates in the algorithm. The results are independent of the isolated domain, according to experimental evidence (at least for the implemented algorithms); that is, when two domains are present, the complexity of each domain is nearly the same whether it is isolated or considered "the remaining part of The DK Design Suite, a system-level algorithm, is first constructed and synthesized at the net-list level. The hardware complexity (resources) at the system level is determined by the corresponding number of NAND gates in the architecture. Clearly, these outcomes are platform-independent. Even if the synthesized designs are eventually targeted to appropriate hardware (using Xilinx ISE software), the resources are only evaluated at the system level. The domain is synthesized and compiled separately at the system level to determine the corresponding number of NAND gates when it is detached from an algorithm. The complexity (i.e., the equivalent number of NAND gates) of the remaining algorithm may be readily computed by subtracting the number of gates in the isolated domain from the overall number of gates in the algorithm. The results of an algorithm, at least for implemented algorithms, are independent of which domain is isolated, according to experimental evidence. In other words, when two domains are present, the complexity of each domain is nearly the same whether it is isolated or referred to as "the remaining part of the

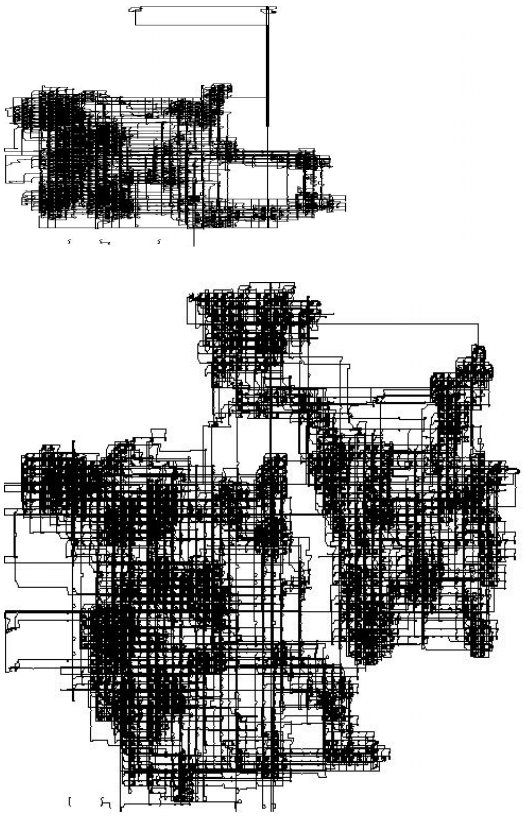

Fig.3.DesignBwithonlydecompressor(15MHz) – Huffman coding.

Processingtimeestimates

7. At the system level, the processing time of a particular algorithm (or its main) is also assessed using the debugging tools in the DK Design Suite. A clock cycle is the basic unit of time measurement. Assuming a parallel run of the domains, the overall processing time is determined by the longer execution time (of the isolated domain or of the remaining method). Estimates of power usage Dynamic power consumption in FPGA is directly influenced by the hardware resources. We presume that the system-level complexity (i.e., the equivalent number of NAND gates) multiplied by the clock frequency describes the dynamic power consumption, which is expressed in NDU (non-defined units). The validity of this approach has been shown by the experiment described in Section 4.2. It should be noted that these power attributes are platform independent.

7.1. Algorithm Partitioning into Parallel Domains–Implementation

To overcome some of the DK Design Suite's shortcomings, we use samples of thirty-two items and arithmetic coding sequences of four symbols. Given that they reflect one second of data gathered by certain sensors (for instance, magnetometers used in wireless sensor networks usually sample at a frequency of 10–50 Hz; see [44]–[47]), these figures make sense. Additionally, the breadth of the processed data is randomly fixed at 10 bits, which is the typical resolution of wire-free sensor networks' analog-to-digital converters (ADCs) [48]. By incorporating the memory required by data reduction techniques within the FPGA, large capacitances of external connections are avoided. Our approach doesn't distort the results since we just use the FPGA-based memory for the tasks that are required and don't store a lot of input or output data samples.

Huffmancoding

The Huffman coding compressor is composed of the following parts: Build-HuffTree, which creates the Huffman tree, BuildHuffCode, which creates the While the latter is done for each new sample, each new code must be decoded into a symbol. Thus, they are in different clock realms. We also decided to store the input data statistics (AlphArray) and internal node structures of the binary tree (InterNode-Array) in memory in the same clock domain as CodeGet. As a result, BuildHuffTree needs to access AlphArray and InterNode-Array via channels. Block diagrams of the Huffman coding decompressor's clock do-main partitioning are shown in Figure 6. The system-level aspects of the design are described in Table 5. primary clock domain The secondary clock domain is called CodeSendDirect.

Channels in the ConstructHuffTreeBuildHuffCodeSampleArray An example of an array channels that use symbol codes SymbolCodeFigure 5: Block schematic of the Huffman coding compressor. Table 4: Processing time and hardware requirements for Huffman coding (compressor). [The same as NAND gates] Cycles of the clock: Domain of the main clock 79,195 1,155 Compressor total: 214,634 Secondary clock domain: 135,439,20,352 The Huffman code, Ding, and CodeSendDirect (computer programming) characteristics (symbols). BuildHuffTree and BuildHuffCode are executed for every new sample, while CodeSendDirect is executed for every new symbol that has to be encoded. Therefore, we decided to put BuildHuffTree and BuildHuff-Code in one clock domain and CodeSendDirect in another. We also decided to create a memory to store the symbol code table (SymbolCode; for symbols encoding) and input data samples (SampleArray) in the same clock domain since CodeSendDirect uses the data principally. Consequently, Build-building a probabilistic model of the sample data), vasCDFCount (building a cumulative distribution function based on the probabilistic model of the sample data), and vCodeEncSeq (encoding the alphabet symbols or sequences of symbols). Every new symbol or symbol sequence that has to be encoded is run using vCodeEncSeq (which is in the opposite clock domain) and every new sample is run using vasPrbCount and vasCDFCount (which are in the same clock domain). The memory that stores a sample of input data (uiaSample), the probabilistic model of input data (asPrb), and the cumulative distribution function of input data (asCumDistFun) is constructed using the same clock domain as vCodeEncSeq. Therefore, in order to access uiaSample, asPrb, and asCumDistFun, VasPrbCount and vasCDFCount need to utilize channels. Block diagrams of the arithmetic coding compressor's clock domain partitioning are shown in Figure 7. The design features are shown in Table 6.

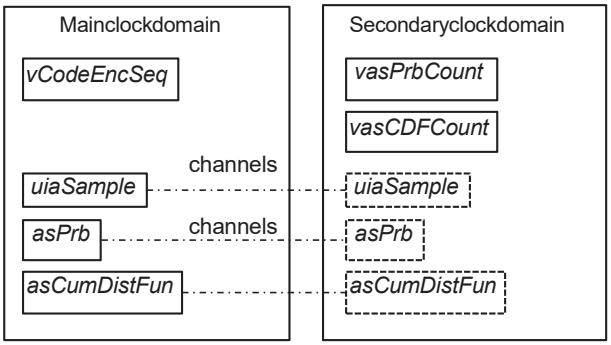

Fig.7.BlockdiagramofArithmeticcodingcompressor.

Table 6. Arithmetic coding (compressor) – hardware resourcesandprocessingtime.

	[NANDgates equivalent]	Clock cycles
Complete compressor	231,666	-

Main clock domain	225,047	3,961
Secondary clock domain	6,619	5,350

The decompressor of our arithmetic coding system consists of vasCDFCount, which constructs the cumulative distribution function based on the probabilistic model of sample data, and vCodeDecSeq, which decodes alphabet symbols or symbol sequences. The first function is executed in response to each new sample, while the latter function is executed in response to each new code that has to be decoded into a symbol or set of symbols. Therefore, we decide to give each function a different clock domain. Furthermore, vCodeDecSeq is implemented in the same clock domain as the memory that houses the probabilistic model of input data (asPrb) and the cumulative distribution function of input data (asCumDistFun). Therefore, in order to access asPrb and asCumDistFun, VasCDFCount has to utilize channels. Block diagrams of the clock domain partitioning of the arithmetic coding decompressor are shown in the figure. 8. The design's attributes are shown in Table 7.

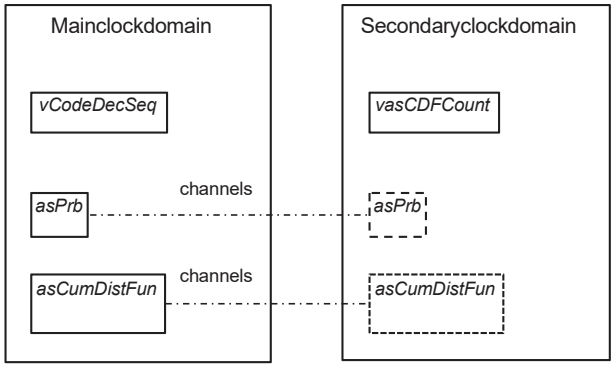

Fig.8.BlockdiagramofArithmeticcodingdecompressor.

Table 7. Arithmetic coding (decompressor) – hardware resourcesandprocessingtime.

	[NANDgates equivalent]	Clock cycles
Complete compressor	303,114	-
Main clock domain	299,679	3,418
Secondary clock domain	3,435	3,204

Channelsoverhead

A partitioned system's resource estimations might be erroneously influenced by the hardware needed for inter-domain communication. To ascertain the actual effect of these overheads, we have created the proper designs using only the channels (in practice, redundant channels that can transmit data samples of 32, 128, and 512 elements are implemented). The results, or the equivalent numbers of NAND gates, are shown in Tables 8 and 9.

Table8.Huffmancoding–channeloverheads.

Sample size	32	128	512
Compressor [NANDgatesequivalent]	216	228	240
Decompressor [NANDgatesequivalent]	680	764	848

Table9.Arithmeticcoding–channeloverheads.

Sample size	32	128	512
Compressor [NANDgatesequivalent]	216	228	240
Decompressor [NANDgatesequivalent]	680	764	848

A partitioned system's resource estimations might be erroneously influenced by the hardware needed for inter-domain communication. To ascertain the actual effect of these overheads, we have created the proper designs using only the channels (in practice, redundant channels that can transmit data samples of 32, 128, and 512 elements are implemented). The results, or the equivalent numbers of NAND gates, are shown in Tables 8 and

Algorithm Partitioning into Parallel Domains–Analysis

8. Tables 4 and 5 (Huffman coding) and 6 and 7 (Arithmetic coding) show the results of algorithm partitioning (using a two-domain partitioning). The nominal clock frequency for the whole design is determined by the longer processing time of a domain in both algorithms (depending on the maximum allowed processing time that cannot be exceeded). According to Equations 1 and 2, any decrease in the clock frequency to a single domain would also result in a decrease in the dynamic power. As a result, we can calculate how much power can be saved by slowing down the other domain, which needs less clock cycles to function.

Huffmancoding

Table 4 demonstrates that whereas the secondary domain requires 20,352 clock cycles to operate (see to Figure 5 for domain details), the primary domain only requires 1,155 cycles. When the compressor design is not partitioned and both domains are driven by the same clock frequency, the total power consumption may be computed (in specific non-defined units (NDU)) as follows: 214,634NDU (79,195 + 135,439) × 1

The partitioned architecture's principal domain, however, may function at a frequency that is just 5.67% of the nominal clock frequency (1,155/20,352 = 0.0567) and yet complete its duty in the same amount of time. As a result, the power consumption of the primary domain may be reduced to: 0.0567 × 79,195 = 4,490.36 NDU while the secondary domain requires: 1 × 135,439 = 135,439 NDU Consequently, the partitio-ned design's overall power consumption equals: 139,929.36NDU = 4,490.36 + 135,439 This is 65.19% of the non-partitioned design's initial 214,634NDU. It saves over 35% of the dynamic power. Applying the same methodology to the decompressor (see Table 5 and Figure 6 for domain details), we find that the nominal clock frequency (655/14,666 = 0.0447) should drive 84,987 equivalent gates of the secondary domain, while 45,737 gates of the main domain only need 4.47% of that frequency. The power usage of the partitioned architecture may thus be expressed as follows: 84,987 × 1 + 45,737 × 0.0447 = 87,031.44NDU

9. 11. This is 66.58% of the power needed for the non-partitioned option, which is 130,724NDU. Arithmetic coding Table 6 demonstrates that 225,047 gates of the major domain may be driven by 74.04% of the nominal frequency (3,961/5,350 = 0.7404) using the same technique for the Arith-metic coding compressor (domain details in Figure 7). In contrast, 6,619 gates of the secondary domain should be driven by the nominal clock. As a result, the nominal clock-powered non-partitioned algorithm consumes the following total power: (225,047 + 6,619) × 1 = 231,666NDU while the anticipated total power of the partitioned design is: 225,047 × 0.7404 + 6,619×1 = 173,243.80NDU As a result, 25.22% less power was used than with the non-partitioned design. The main domain, which has 299,679 gates, sets the nominal clock frequency for the Arithmetic

Coding Decompressor (details in Table 7 and Figure 8). The secondary domain, which has only 3,435 gates, needs 93.74% of the frequency. The power savings in this case are minimal, i.e. (299,679 + 3,435) × 1 = 303,114NDU as opposed to 299,679 × 1 + 3,435 × 0.9374 = 302,898.97NDU The loss of dynamic power is just 0.07%.

10. RemarksonSequentialPartitioning

Algorithms in which each component runs simultaneously, although perhaps at separate clock speeds or with various intensities, may be implemented using the basis for algorithm partitioning discussed in Section 4. However, this approach may not save enough power in certain situations (such with the Arithmetic coding de-compressor). Because both parts of the method need almost the same amount of processing time, as seen in Table 7, we cannot expect any significant power savings from parallel partitioning, independent of domain sizes. However, there are a number of algorithms where processing is at least somewhat sequential and where it is not necessary to run every component of a deconstructed method at the same time. Since the dynamic power consumption depends on the switching activities of the relevant resources, idle fragments or those with temporarily very low switching activity utilize comparatively less dynamic power. By using this feature, more system-level dynamic power savings are possible. Consider an algorithm that can be broken down into only two parts, X and Y, that are executed one after the other. Assume that the corresponding domains Dx and Dy have processing times of cx and cy clock cycles, respectively. The whole algorithm execution time may thus be expressed as follows:

where the system-level estimated designs for fx and fy are the equivalent domain clocktitioned designs, in ge- frequencies. Theoretically, power consumption might be reduced by reducing the clock frequency of the more hardware-intensive domain and increasing the clock frequency of the other domain proportionately (if preserving the system's overall throughput is necessary). Since changing the starting frequencies (by Δfx and Δfy, respectively) would change the execution time overall, Equation 3 may be utilized to simply prove the following dependency: neral, even if the actual layouts of the implemented designs differ, inherited at the hardware level. Thus, experimental evidence has been used to support the validity of the suggested strategies.
Our research focuses on data reduction algorithms, namely arithmetic and Huffman coding. As a result, some of these algorithms' characteristics have also been (intentionally or not) made public. Specifically, we discovered that arithmetic coding is a viable option for FPGA-based data reduction, which goes against the conventional wisdom.

1. Devices will be used in energy-conscious systems such as wireless sensor networks. Despite a comparatively high static power (the Xilinx Vir- has a static power of 378mW),

The recommended frequencies (i.e., reducing the overall power consumption) may then be found by optimizing hardware-frequency products within the limitations given by (5). The proposed approach ignores many practical issues. First, the statement that there is no dynamic power during the idle periods is only partially accurate. Second, the static power inherent in all FPGA devices may further skew the accuracy of computations. Consequently, experimental validation is necessary to validate the feasibility of this strategy for the real reduction in power consumption. The tests are now being conducted, and the results will be reported in our next publications.

11. Conclusions

In this study, we proposed methods for optimizing the dynamic power consumption of

FPGA devices at the system level. Algorithm decompositions into simultaneously executed chunks have been calculated and experimentally shown to significantly reduce the dynamic power when paired with the appropriate clock frequency selection. Furthermore, we suggest a sequential algorithm partitioning as an extra/alternative technique that might further reduce the power consumption by altering the clock frequencies of the relevant clock domains (without compromising the algorithm's

throughput). It is important to note that the proposed principles do not add any unintentional processing, delays, or major hardware overheads. Our power-saving estimates are purposefully based only on system-level outcomes. Thus, with a variety of FPGAs and other devices, the dynamic power savings ought to be consistent. Clearly, the actual numbers in milliwatts will not be device-independent; only the dynamic-to-static power ratio will be. tex-II FPGA), they provide several benefits, such as dynamic power minimization if the design is big or at least modest. For instance, up to 533mW of dynamic power may be used by a design that only uses one-third of Virtex-II FPGA slices. Therefore, it seems that our attempts to reduce dynamic power are not without merit.

Lastly, a significant path for the outcomes in the future might be indicated. Algorithm partitioning was carried out intuitively in the trials (based on our knowledge of the structure of the algorithm). We think system-level algorithm partitioning for power consumption optimization is an intriguing issue, despite the fact that it presently seems to be the most common (and practical) option. Our tests clearly show that a suggested parallel partitioning method is, for instance, to divide into domains with opposing attributes (big domains with slow clocks against tiny domains with extremely rapid clocks). Furthermore, we discovered that, in contrast to popular belief, inter-domain communication resources in FPGA implementations are usually negligible when compared to the size of (moderate and large) de-signs. As further study is done in this area, other intriguing features may become apparent.

References

[1] K. Romer,F. Mattern, ―The Design Space of Wireless Sensor Networks,‖ IEEE Wireless Communications, vol. 11, no. 6, December 2004, pp. 54-61.

[2] M.A.M.Vieira,C.N.Jr.Coelho,D.C.Jr.daSilva,J.M. da Mata, ―Survey on Wireless Sensor Network Devices‖.In:ProceedingsoftheEmergingTechnologiesand FactoryAutomation,2003,pp.537-544.

[3] J.Feng,F.Koushanfar,M.Potkonjak,―System-Archi- tectures for Sensor Networks Issues,Alternatives, and Directions‖.In:ProceedingsoftheIEEEInternational Conference on VLSI in Computers and Processors, 2002,pp.226-231.

[4] Xilinx, Inc., ―Product Selection Guides,‖ FPGA and CPLDSolutionsfromXilinx,Inc.,2008.[Online].Available:http://www.xilinx.com.[Accessed:September06, 2008].

[5] Altera Corporation, ―Altera Product Catalog,‖ Altera - FPGA, CPLD, ASIC and Programmable Logic, 2008. [Online].Available:http://www.altera.com.[Accessed: September06,2008].

[6] LatticeSemiconductorCorporation,―LatticeMicoPro- ductBrochure,‖FPGAandCPLDsolutionsfromLattice Semiconductor,2008.[Online].Available:http://www. latticesemi.com.[Accessed:September06,2008].

[7] Tensilica,Inc.,―XTensaProductBrief,‖Tensilica:Con- figurable and Standard Processor Cores for SOC Design,2008. [Online].Available: http://www.tensilica. com.[Accessed:September06,2008].

[8] B.O'Flynn,etal.,―TheDevelopmentofaNovelMini- aturized Modular Platform for Wireless Sensor Networks,‖inProceedingsoftheFourthInternationalSym- posiumonInformationProcessinginSensorNetworks, 2005,pp.370-375.

[9] S.J. Bellis, K. Delaney, B. O'Flynn, J. Barton, K.M. Razeeb, C. O'Mathuna, ―Development of Field Programmable Modular Wireless Sensor Network Nodes forAmbientSystems,‖ComputerCommunications,vol. 28, no. 13,August 2005, pp. 1531-1544.

[10] D.Bauer,S.Furrer, S.Rooney,W.Schott, H.L.Truong, B.Weiss,―TheZRLWirelessSensorNetworkingTest- bed,‖ IBM Zurich Research Laboratory, Ruschlikon, Switzerland,Tech. Rep. RZ 3620 (# 99630), 2005.

[11] V.Tsiatsis,S.A.Zimbeck,M.B.Srivastava,―Architec- tureStrategiesforEnergy-EfficientPacketForwarding

inWirelessSensorNetworks‖.In:ProceedingsoftheIn- ternationalSymposiumonLowPowerElectronicsand Design,2001,pp.92-95.

[12] L. Shang, A.S. Kaviani, K. Bathala, ―Dynamic Power ConsumptioninVirtex-IIFPGAFamily‖.In:Proceedingsofthe2002ACM/SIGDA10th InternationalSympo- sium on Field-Programmable Gate Arrays, 2002, pp. 157-164.

[13] V. Degalahal, T. Tuan, ―Methodology for High Level Estimationof FPGAPower Consumption‖. In:Proceedingsofthe2005ConferenceonAsiaSouthPacificDe- signAutomation,2005,pp.657-660.

[14] N. Rollins, M.J. Wirthlin, ―Reducing Energy in FPGA MultipliersThrough Glitch Reduction,‖ presented at the International Conference on Military and Aerospace ProgrammableLogicDevices,Washington,DC,USA, 2005.Celoxica,Ltd.,―AgilityCompiler,‖Celoxica-TheTech- nology Leader in C Based Electronic Design and Syn- thesis,2006.[Online].Available:http://www.celoxica. com/products/agility/default.asp. [Accessed: October 18,2006].

[15] S.J.E.Wilton,S.-S.Ang,W.Luk,―TheImpactofPipe- liningonEnergyperOperationinField-Programmable GateArrays‖.In:FieldProgrammableLogicandAppli- cation, Vol. 3203, J. Becker, M. Platzner, and S. Vernalde,Eds.Berlin,Germany:Springer-Verlag,2004, pp.719-728.

[16] G.J.M.Smit,P.J.M.Havinga,―ASurveyofEnergySa- vingTechniquesforMobileComputers,‖Universityof Twente, Department of Computer Science, Enschede, Netherlands,Tech.Rep.Moby Dick,1997.

[17] P.J.M.Havinga,G.J.M.Smit,―LowPowerSystemDe- signTechniquesforMobileComputers,‖Universityof Twente, Department of Computer Science, Enschede, Netherlands,Tech.Rep.ISSN 1381-3625, 1997.

[18] O.S.Unsal,I.Koren,―System-LevelPower-AwareDe- sign Techniques in Real-Time Systems‖. In: Proceed- ings of the IEEE, vol. 91, no. 7, July 2003, pp. 1055-1069.

[19] H.G.Lee,S.Nam,N.Chang,―Cycle-AccurateEnergy MeasurementandHigh-LevelEnergyCharacterization of FPGAs‖. In: Proceedings of the 4th International Symposium on Quality Electronic Design, 2003, pp. 267-272.

[20] N.Chang,K.Kim,―Real-Timeper-CycleEnergyCon- sumption Measurement of Digital Systems‖, ElectronicsLetters,vol.36,no.13,June2000,pp.1169-1171.

[21] K.Weiß,C.Oetker,I.Katchan,T.Steckstor,W.Rosen- stiel, ―Power Estimation Approach for SRAM-based FPGAs‖, In: Proceedings of the 2000 ACM/SIGDA 8th InternationalSymposiumonFieldProgrammableGate Arrays,2000,pp.195-202.

[22] M.French,―APowerEfficientImageConvolutionEn- gineforFieldProgrammableGateArrays‖.In:InternationalConferenceonMilitaryandAerospaceProgram- mable Logic Devices,Washington, DC, USA, 2004.

[23] A.Deligiannakis,Y.Kotidis,N.Roussopoulos,―Com- pressing Historical Information in Sensor Networks‖. In: Proceedings of the 2004 ACM SIGMOD Interna- tional Conference on Management of Data, 2004, pp. 527-538.

[24] M.Chen,M.L.Fowler,―TheImportanceofDataCom- pressionforEnergyEfficiencyinSensorNetworks‖.In: Proceedings of the 2003 Conference on Information SciencesandSystems,2003.

[25] M.Chen,M.L.Fowler,―DataCompressionTrade-Offs inSensorNetworks‖.In:ProceedingsoftheSPIECon- ference on Mathematics of Data/Image Coding, Com- pression,andEncryptionVII,withApplications,2004, pp.96-107.

[26] A. Deligiannakis, Y. Kotidis, ―Data Reduction Tech- niques in Sensor Networks‖, IEEE Data Engineering Bulletin, vol. 28, no. 1, March 2005, pp. 19-25.

[27] K. Sayood, Introduction to Data Compression, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2006.

[28] D. Salomon, Data Compression - The Complete Refe- rence, 4th ed. London, UK: Springer-Verlag, 2007.

[29] T.Dang,N.Bulusu,W.Feng,―RIDA:ARobustInfor- mation-DrivenDataCompressionArchitectureforIrre- gular Wireless Sensor Networks‖. In: Wireless Sensor Networks,vol.4373,K.Langendoen,T.Voigt,Eds.Ber- lin,Germany:Springer-Verlag,2007, pp. 133-149.

[30] V.Jolly,S.Latifi,N.Kimura,―Energy-EfficientRouting inWirelessSensorNetworksBasedonDataReduction‖. In:ProceedingsoftheInternationalConferenceonPa- rallelandDistributedProcessingTechniquesandApplications,2006,pp.804-812.

[31] N.Kimura,S.Latifi,―ASurveyonDataCompressionin Wireless Sensor Networks‖. In: Proceedings of the InternationalConferenceonInformationTechnology:Co- dingand Computing, 2005, pp. 8-13.

[32] K.C.Barr,K.Asanovic,―Energy-AwareLosslessData Compression,‖ ACM Transactions on Computer Sys- tems, vol. 24, no. 3, pp. 250-291,August 2006.

[33] C.M. Sadler, M. Martonosi, ―Data CompressionAlgo- rithms for Energy-Constrained Devices in DelayTolerantNetworks‖.In:Proceedingsofthe4th International Conference on Embedded Networked Sensor Systems, 2006,pp.265-278.

[34] J.P. Lynch,A. Sundararajan, K.H. Law,A.S. Kiremid- jian,E.Carryer,―Power-EfficientDataManagementfor aWirelessStructuralMonitoringSystem‖.In:Proceed- ingsoftheFourthInternationalWorkshoponStructural HealthMonitoring,2003,pp.15-17.

[35] H.Akcan,H.Bronnimann,―DeterministicDataReduc- tion in Sensor Networks‖. In: Proceedings of the 2006 IEEE International Conference on Mobile Adhoc and SensorSystems,2006,pp.530-533.

[36] A.T. Hoang, M. Motani, ―Collaborative Broadcasting andCompressioninCluster-basedWirelessSensorNet- works‖.In:ProceedingsoftheSecondEuropeanWork- shoponWirelessSensorNetworks,2005,pp.197-206.

[37] P.J.Marron,R.Sauter,O.Saukh,M.Gauger,K.Rother- mel, ―Challenges of Complex Data Processing in Real WorldSensorNetworkDeployments‖.In:Proceedings of theACM Workshop on Real-World Wireless Sensor Networks,2006,pp.43-48.

[38] D. Petrovic, R.C. Shah, K. Ramchandran, J. Rabaey, ―Data Funneling: Routing withAggregation and Com- pressionforWirelessSensorNetworks,‖inProceedings of the First 2003 IEEE International Workshop on Sensor Network Protocols andApplications, 2003, pp. 156-162.

[39] A. Deligiannakis, Y. Kotidis, ―Data Reduction Tech- niques in Sensor Networks‖, IEEE Data Engineering Bulletin, vol. 28, no. 1, March 2005, pp. 19-25.

[40] Y. Al-Obaisat, R. Braun, ―On Wireless Sensor Net- works:Architectures,Protocols,Applications,andMa- nagement‖.In:Proceedingsofthe1st IEEEInternational ConferenceonWirelessBroadbandandUltraWidebandCommunication,2006.

[41] A. Ciancio, S. Pattem, A. Ortega, B. Krishnamachari, ―Energy-EfficientDataRepresentationandRoutingfor WirelessSensorNetworksBasedonaDistributedWa- velet CompressionAlgorithm‖. In: Proceedings of the Fifth International Conference on Information Pro- cessing in Sensor Networks, 2006, pp. 309-316.

[42] P.P. Czapski, A. Sluzek, ―Power Optimization Tech- niquesin FPGADevices:ACombination of System- and Low-Levels,‖ International Journal of Electrical, Com-puter,andSystemsEngineering,vol.1,no.3,2007,pp. 148-